Loop-Free Internet Routing Using Hierarchical Routing Trees

Shree Murthy[2]
Sun Microsystems, Inc.
2550 Garcia Avenue
Mountain View, CA 94043

J.J. Garcia-Luna-Aceves
Computer Engineering Department
University of California
Santa Cruz, CA 95064

Abstract

We present a new hierarchical routing algorithm that combines the loop-free path-finding algorithm (LPA) with the area-based hierarchical routing scheme first proposed by McQuillan for distance-vector algorithms. The new algorithm, which we call the Hierarchical Information Path-based Routing (HIPR) algorithm, accommodates an arbitrary number of aggregation levels and can be viewed as a distributed version of Dijkstra's algorithm running over a hierarchical graph. HIPR is verified to be loop-free and correct. Simulations are used to show that HIPR is much more efficient than OSPF in terms of speed, communication and processing overhead required to converge to correct routing tables. HIPR constitutes the basis for future Internet routing protocols that are as simple as RIPv2, but with no looping and better performance than protocols based on link-states.

1. Introduction

Routing information maintained at each router has to be updated frequently to adapt to network dynamics. In a flat routing architecture, the size of the routing tables grows linearly with the number of destinations in the network. Accordingly, aggregation of routing information becomes a necessity in any type of routing protocol. Hierarchical routing for datagram computer networks was first proposed by McQuillan [10]. According to this scheme (analyzed by Kamoun and Kleinrock [7]), the nodes in the network are organized into clusters or areas by grouping (clustering) together the nodes which are close by. Each of these areas are single addressable entities from the point of view of higher level areas. The topology within the area is transparent to the nodes outside the area. The distance from a source node to an area represents the actual distance in physical hops (number of nodes traversed) from the source node to the destination remote area.

There have been several subsequent proposals for hierarchical routing, which vary in the way in which the nodes are organized (addressing scheme) and the routing algorithms used [1], [15], [4], [8], [16]. With very few exceptions [5], prior proposals for hierarchical routing have assumed variants of the Distributed Bellman-Ford (DBF) algorithm or topology-broadcast algorithms. In the backbone scheme for hierarchical routing used in OSPF [11], a network is divided into areas connected by a backbone.

[1] This work was supported by the Defense Advanced Research Projects Agency (DARPA) under contract DAAB07-95-C-D157

[2] This work was done at UC Santa Cruz

A number of algorithms have been proposed [2], [6], [13], [14] that eliminate the counting-to-infinity problem of DBF and eliminate loops altogether [3]. These algorithms are based on the maintenance and incremental exchange of shortest-path routing trees. These can be viewed as distributed implementation of Dijkstra's shortest path algorithm, where as link-state algorithms require a router to have a topology map over which it runs Dijkstra's algorithm. A few of these shortest-path-tree-based algorithms have been shown to outperform link-state algorithms [3], [13]. However, because they rely on each router knowing the shortest path tree to every destination, they force a router to know more "host routes" (corresponding to individual routers) than would be necessary if DBF were used. Furthermore, in the Internet, routing information for individual remote routers may not be available because of subnetting and masking.

This paper presents, verifies, and analyzes the performance of the first hierarchical routing algorithm based on the maintenance and exchange of hierarchical routing trees. We call this algorithm the *Hierarchical Information Path-based Routing* (HIPR) algorithm. The main idea of HIPR is to provide a distributed implementation of Dijkstra's shortest path algorithm running over a hierarchical graph organized in areas according to McQuillan's scheme for hierarchical routing. The loop-free path-finding algorithm (LPA) [3] is extended for this purpose.

HIPR, which implements a distributed version of Dijkstra's algorithm, is compared with OSPF, which is based on replicated implementation of Dijkstra's algorithm. The simulation results clearly show that HIPR is far superior to OSPF in terms of number of steps, overhead traffic, and processing load required to update routing tables. HIPR constitutes the basis for new internet routing protocols that are as simple as RIPv2 [9] and is much more efficient than OSPF or any other routing protocol based on topology broadcast.

2. Internetwork Model

An internetwork is modeled as an undirected connected graph in which the routing nodes are the nodes of the graph and direct links between the routers are the edges of the graph. Each node represents a router and is a computing unit involving a processor, local memory and input and output queues. Each link in such a graph has two costs, one in each direction, associated with it. A link exists in both directions at any one time. The lower-level protocol ensures reliable delivery of packets. All messages are processed one at a time in the order in which there are received. Link

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
1997		**00-00-1997 to 00-00-1997**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Loop-Free Internet Routing Using Hierarchical Routing Trees	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Santa Cruz,Department of Computer Engineering,Santa Cruz,CA,95064**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**8**	

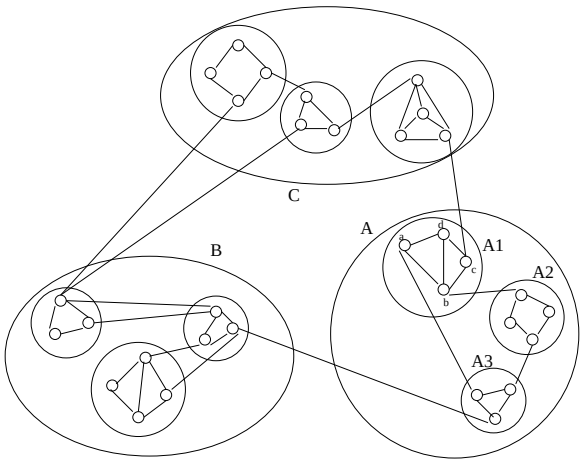

Fig. 1. Example of the Hierarchical Network Topology

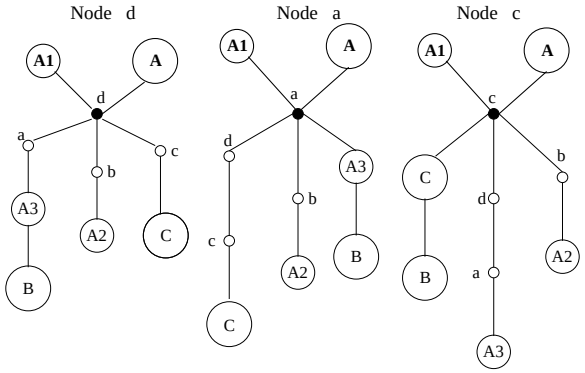

Fig. 2. Hierarchical Routing Trees at Nodes

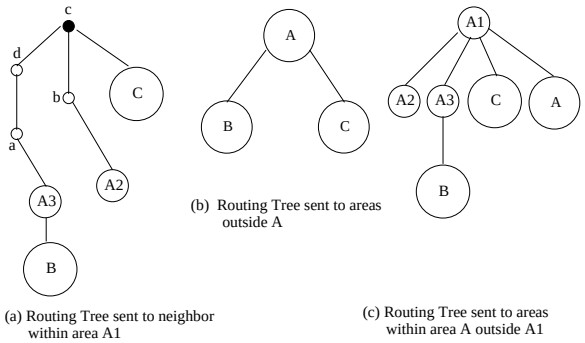

Fig. 3. Hierarchical Routing Trees Sent by Border Nodes

costs vary in time but are always positive. The cost of the failed link is marked as infinity. A node failure is modeled as all links incident on that node failing at the same time.

Following the McQuillan's approach to area-based routing, hosts and networks can be organized into L levels of areas. An area at level k is called a k-area; routers form 0-area. A group of nodes forms a 1-area and a group of k-areas form a $k+1$-area. Each router can belong to only one area at each level in the hierarchy. A *boundary node* or a *border node* is defined as a router with direct connectivity with the boundary nodes in other areas. The distance from a source router to a remote k area represents the length of the true shortest-path from a node to the remote area. Clearly, the distance from a router to another router in the same 1-area is the true shortest path distance. Similarly, the distance from a router to itself (or any directly adjacent host) is 0 and the distance from the router to its own k-area is 0.

In OSPF terminology, a routing node connected to the backbone network serves as a boundary node. Routers not connected to the backbone are simple nodes. By taking into account destination behind routers, masks and subnets, our description of the network hierarchy can be mapped into a hierarchical addressing scheme that would be practical on an Internet. For our purposes, it suffices to illustrate the fact that in HIPR, a destination is a single entity or an arbitrary aggregation of entities following McQuillan's scheme.

3. HIPR

3.1 Design Principle

The basic design concept of HIPR is simple. Each router communicates to its neighbors its hierarchical routing tree in an incremental fashion. Its hierarchical routing tree consists of all its preferred hierarchical shortest paths to each known destinations in its own 1-area, all known highest-level areas, in general all $(k-1)$-areas within its own k-area. This implies that border nodes forbid detailed routing tree information regarding their own areas from percolating to other areas. Hence, for a hierarchical routing tree, an entire foreign area is simply another node in the tree. Figures 1 and 2 illustrates this idea. Figure 1 shows the topology of a network organized into three hierarchical levels; Figure 2 shows the hierarchical routing tree that router d needs to communicate to its neighbor routers (within area $A1$) and the hierarchical routing trees of its two neighbors a and c. Bold nodes in the figure corresponds to "self reference" entries in the router's routing table. Notice that the path from d to remote areas consists of both routers within router d's 1-area and other remote areas.

Consider a border node in the given topology (say c). The hierarchical routing tree propagated to all nodes within the area containing node c is shown in Figure 3(a). For the areas outside the area containing node c (i.e., outside A), the hierarchical routing tree sent to the peer nodes of other areas is shown in Figure 3(b).

Consider border node b in Figure 1. This is a border node connecting areas $A1$ and $A2$ which are contained in the bigger area A. The routing tree sent by node b to its peer border node in area $A2$ (within area A) is shown in Figure 3(c).

Routers exchange their hierarchical routing trees incrementally by communicating only the hierarchical distance and second-to-last hop (predecessor) to each destination. In the case of destinations within router i's 1-area, the second-to-last hop consists of a router.

In the case of a remote area known to router i, the predecessor consists of either a border node in router i's 1-area or a remote area. The hierarchical distance to a destination consists of the true shortest distance to a border node. The rest of this section describes the information and procedures that HIPR uses to update the hierarchical routing trees of routers, examining that no routing table loops are ever formed. In essence, HIPR uses a distributed implementation of Dijkstra's shortest-path algorithm over a hierarchical graph.

Internet routing protocols maintain routing information about networks (not individual hosts) and must support subnetting. To use HIPR as a scalable Internet routing protocol, it should support a minimum of one level of areas. Even though HIPR supports multiple levels of areas, because OSPF supports only two levels, in this paper we discuss HIPR based on only one level of areas.

3.2 Information Maintained at a Router

Each router maintains a single routing table that can be logically divided into two parts: a *node-level routing table* (NRT) and a *area-level routing table* (ART). The NRT portion of the routing table maintains information about the routers and destinations in the same area with which a node is affiliated. The ART portion of the routing table maintains information about other areas (higher-level areas to which a node belongs). Both parts of the routing table are updated using the same algorithm.

The entry for destination j in node i's NRT consists of a destination's identifier, the distance to the destination (D_j^i), the successor (s_j^i), the predecessor (p_j^i) along the preferred path (shortest path) to the destination, and the feasible distance to the destination (FD_j^i), which we define subsequently. The NRT also maintains a marker (denoted by tag_j^i) used to update the routing table entries. For destination j, tag_j^i specifies whether the entry corresponds to a simple path (tag_j^i = correct), a loop (tag_j^i = error) or a destination that has not been marked (tag_j^i = null). This marker is used to reduce the number of routing table entries that need to be processed after each input event impacting the routing table.

The ART maintains similar information for each area known to node i. The distance from a source to destination area is the length of the hierarchical path from the source to the destination area. The successor entry for the ART is the *next area* in the path towards the destination area and the predecessor entry is the *area* previous to the destination area. The successor and the predecessor entries will be *null* when i determines that the individual or area destination is unreachable.

In addition to the routing table, a router also maintains a *distance table*, *link-cost table* and a *reply-status table*. Distance table maintained at each router is a matrix containing for each destination j, the hierarchical routing tree reported by its neighbors $k \in N_i$. At router i, D_{jk}^i and p_{jk}^i represents the distance and the predecessor reported by neighbor k for destination j, respectively.

The link-cost table contains the cost of each link incident to the node maintaining the table. The cost of the link is denoted by l_{ik}. The cost of an inactive link is set to infinity.

The reply-status table at node i maintains the values of the reply status flag (r_{jk}^i) for each neighbor, k, and for each known destination and area, j, to node i. Each entry in this table indicates whether or not a node is waiting to get a reply from its neighbor in response to its query.

3.3 Information Exchanged between Nodes

Routing information is exchanged among neighboring nodes by means of update messages. An update message from router i consists of a vector of entries reporting incremental updates of its routing table; each entry specifies an update flag (denoted by u_j^i), a destination j (i.e., an individual node or an area), the reported distance to that destination, and the reported predecessor (individual node or an area) in the path to the destination. The update flag indicates whether the entry is an update ($u_j^i = 0$), a query ($u_j^i = 1$), or a reply ($u_j^i = 2$).

Because every router reports to its neighbors the predecessor in the shortest path to the destination, the complete hierarchical path to any destination (called the implicit path to the destination) is known by the router's neighbors. This is done by the path traversal on the predecessor entries reported by the router [12].

In the specification of HIPR, the successor to a destination j for any router is simply referred to as the successor of the router. Same reference applies to other information maintained by the router. Similarly, updates, queries and replies refer to destination j unless otherwise stated.

The information propagated to the neighbor depends on whether the node is a border node or not. Border nodes block any information about its local area before sending an update to other border nodes (in its peer areas). This ensures that the algorithm is scalable.

A router can be in one of the two states *active* or *passive* at any given time. A router i initializes itself in passive state with an infinite distance for all its known neighbors and with a zero distance to itself. After initialization, router i sends updates containing the distance to itself to all its neighbors.

3.4 Distance Table Updating

The procedures used in HIPR to update the entries of the routing tables are similar to the procedures used in LPA [3]. The key difference is that, a border node at level k supports hierarchical routing by making sure that no routing information regarding destinations in its own 1-area or any $(k-1)$-areas in its own k-area percolates to a neighbor border node in another k-area.

When router i receives an input event indicating the change in the cost of link (i, k), it updates its link-cost table with the new cost of the link and then updates

the distance table entries making sure that the implicit paths after the changed state of the network does not imply any loops. Updating the distance table entries erases the outdated path information by making the path information consistent with the latest update. This is done by updating the distance and predecessor information for each destination j affected by the input event ($D^i_{jk} = D^k_j + d_{ik}$ and $p^i_{jk} = p^i_k$). In addition to this, the path to any destination j through any other neighbor which includes neighbor k is also updated. This is done by walking through the distance table entries of a node. If the path implied by the predecessor reported by router b ($b \neq k$ and $b \in N_i$) to destination j includes router k, then the distance and predecessor entries are updated for that path.

The topology information within an area is transparent to nodes outside the area. Border or boundary nodes prevent the propagation of routing information outside an area by blocking such messages. All updates received by a node are processed in a similar manner.

3.5 Blocking Temporary Loops

A router forces its neighbors not to use it as a successor (next hop) to a given destination when it detects the possibility of a temporary loop. This is done by using an interneighbor coordination mechanism. The algorithm defines a *feasibility condition* (FC) using which, a complete order of routes along a given path can be established. The *feasible distance* (FD^i_j) of router i for destination j is the smallest value achieved by the router's own distance to j since the last time router i sent a query reporting an infinite distance to j. Between two synchronization points (query originating points), the cost of the link can change or remain the same but cannot increase. This ensures that routing table loops are eliminated.

The feasible distance to a destination is initialized to infinity. At every synchronization point, the feasible distance is taken as the minimum of the existing feasible distance and the shortest path entries. Whenever the feasible distance has to be increased, an interneighbor coordination mechanism is initiated by the exchange of queries and replies.

A router is chosen as a successor to a destination only if it satisfies the following feasibility condition.

Feasibility Condition: At time t, router i can choose any router $n \in N_i(t)$ as its new successor s^i_j such that $D^i_{jn}(t) + d_{in}(t) = D_{min}(t) = \min\{D^i_{jx}(t) + d_{ix}(t) | x \in N_i(t)\}$ and $D^i_{jn}(t) < FD^i_j(t)$. If no such neighbor exists and $D^i_j(t) < \infty$, router i must keep its current successor.

3.6 Routing Table Updating

After the path information is updated, the way in which router i updates its routing table for a given destination depends on whether router i is *passive* or *active* for that destination. A router is passive if it has a *feasible successor*, or has determined that no such successor exists and is active if it is searching for a feasible successor. A feasible successor for router i with respect to destination j is a neighbor router that satisfies FC.

When router i is passive, it reports the current value of D^i_j in all its updates and replies. However, while router i is active, it sends an infinite distance in its replies and queries. An active router cannot send an update regarding the destination for which it is active. This is because an update during active state would have to report an infinite distance to ensure that the inter-neighbor coordination mechanism used in HIPR provides loop freedom at every instant.

If router i is passive when it processes an update for destination j, it determines whether or not it has a feasible successor, i.e., a neighbor router that satisfies FC.

If router i finds a feasible successor, it sets FD^i_j equal to the smaller of the updated value of D^i_j and the current value of FD^i_j. In addition, it updates its distance, predecessor, and successor making sure that only simple paths are used.

Router i then sends updates to its neighbors if its routing table entry changes. Alternatively, if router i finds no feasible successor, then it sets $FD^i_j = \infty$ and updates its distance and predecessor to reflect the information reported by its current successor. If $D^i_j(t) = \infty$, then $s^i_j(t) =$ null. Router i also sets the reply status flag ($r^i_{jk} = 1$) for all $k \in N_i$ and sends a query to all its neighbors. Router i is then said to be *active*, and cannot change its path information until it receives all the replies to its query.

The tagging mechanism used for routing table updating ensures that only those routing table entries affected by the input event will be traversed and updated. i.e., if there is a topology change in the existing routing tree then, only nodes which are downstream to that topological change need to be updated since this change does not affect nodes upstream to the topological change. This mechanism minimizes the processing that has to be done for each update message.

A path $P^i_{jk}(t)$ is defined by the predecessors reported by neighbor k to router j stored in i's distance table at time t. To ensure loop-free paths, a path traversal from j back to k is made using the predecessor information. Complete or partial path can be traversed. The path traversal ends when either a predecessor x for which $tag^i_x = correct$ or $tag^i_x = error$ or neighbor k is reached. If $tag^i_x = error$, then tag^i_j is set to error also; otherwise, the neighbor k or a correct tag must be reached in which case tag^i_j is set to correct. This mechanism ensures loop-free paths without having to traverse the entire routing table.

3.7 Processing of Queries and Replies

Queries and replies are processed in a manner similar to the processing of an update, as described above. If the input event that causes router i to become active is a query from its neighbor k, router i sends a reply to router k reporting an infinite distance. This is the case, because router k's query, by definition, reports the latest information from router k, and router i will send an update to router k when it becomes passive if its distance is smaller than infinity. A link-cost change

is treated as a link failing and recovering with a new link cost.

When router i is active and receives replies from all its neighbors, it resets the reply flag ($r^i_{jk} = 0$). This means that router i's neighbors have processed the query. Therefore, router i is free to choose any neighbor that provides the shortest distance, if there is any. If such a neighbor is found, router i updates the routing table with the minimum distance and sets $FD^i_j = D^i_j$.

If router i is passive and has already set its distance to infinity ($D^i_j = \infty$), and receives an input event that implies an infinite distance to j, then router i simply updates D^i_{jk} and d_{ik} and sends a reply to router k with an infinite distance if the input event is a query from router k. This ensures that updates messages will stop when a destination becomes unreachable.

A router does not wait indefinitely for replies from its neighbors because a router replies to all its queries regardless of its state. Thus, there is no possibility of deadlocks due to the inter-neighbor coordination mechanism.

3.8 Example

Consider the topology in Figure 1. We concentrate on area $A1$ which contains a four-node network. Area $A1$ has three border nodes, a, b and c through which it is connected to areas $A3$ (and B), $A2$ and C, respectively. To simplify the description of how HIPR operates, a "virtual topology" of areas and routers is shown in Figure 4 (a). This topology consists of the destinations known to routers in area $A1$ and the links between destinations that may be known to the routers in area $A1$ depending on their routing trees. This forms a eight-node network, with each node representing either a node or an area. A link to an area indicates the connectivity from a border node outside the area to a border node inside the area. A node indicating an area represents the border node(s) that provides inter-area connectivity to the area. Message sent by an "area node" over a link corresponds to messages sent by the corresponding border node. We explain the working of HIPR on this topology when the link connecting areas A and C fails and focus on the routing table entry for destination C.

In Figure 4, the number adjacent to each link represents the weight of that link; U, Q and R represents updates, queries and replies respectively. The arrowhead from node x to node y indicates that node y is the successor of node x towards destination j; i.e., $s^x_j = y$. The label in the parenthesis assigned to node x indicates current distance (D^x_j) and the feasible distance from x to destination j (FD^x_j).

When link (c, C) fails, node c updates its distance table by setting the distance to area C to ∞. Node c is unable to find a feasible successor which satisfies the FC to reach area C. This is because the distance to C through its other neighbors d and b is greater than the feasible distance (i.e., $D^c_{Cd} > FD^c_C$ and $D^c_{Cb} > FD^c_C$). Accordingly, node c sends a query to all its neighbors (Figure 4 (b)).

When nodes d and b receive the query, they update its distance tables and determine that they have no feasible successor to reach area-node C. In turn, they become active by sending out a query to their neighbors and also reply to node c with an infinite distance (Figure 4 (c)).

Upon receiving the queries about destination C, nodes a and $A2$ reply with a finite distance to their neighbors, because they have a feasible successor satisfying the feasibility condition. Also, node a updates its distance and routing table entries as it now uses a different path to reach C and also sends an update regarding this (Figure 4 (d)). On receiving replies with finite distances, nodes d and b update their distance and routing table entries and in turn send updates about their new path and distance to reach area-node C to all their neighbors (Figure 4 (e)). Finally, node d updates its distance and routing table entries, recomputes its feasible distance and sets node d as its successor node to reach area-node C. Updates are sent accordingly (Figure 4 (f)). Any topological change within an area is transparent to the nodes outside the area. When a link within area $A1$ fails, HIPR will be run within area $A1$ only and this information will not be propagated to nodes outside $A1$.

4. Correctness of HIPR

Loop freedom is guaranteed at all times in topology G if the successor graph $S_j(G)$ is always a directed acyclic graph. In the steady state, when all routing tables are correct, $S_j(G)$ must be a directed tree pointed towards j. The following theorem proves that HIPR is loop free.

Theorem 1: *HIPR is loop-free at every instant.*

Proof: Let G be a stable topology and let the successor graph $S_j(G)$ be loop-free at every instant before time t. No loops can be created at this state unless routers change successors and modify the successor graph.

There can be two instances when a successor graph can change: (a) the successor graph within an area can change, or (b) the successor graph of the virtual network can change.

Because HIPR is the same as LPA within an area, the proof that HIPR is loop-free for case (a) follows from the proof that LPA is loop-free [3]. Because all areas in the network form a virtual heterarchical network, and each area is viewed as a single router/destination from a local node, case (b) reduces to case (a). Therefore, the successor graph is loop-free for case (b) also. This proves Theorem 1. □

To prove that HIPR converges to correct routing table entries, we assume that after a finite time T_c, no more resource changes (link-cost change or topology change) occur. We extrapolate the correctness properties of LPA [3] to prove that HIPR converges to correct routing tables.

Theorem 2: *HIPR is live.*

Proof: Consider the case when the network topology is stable. When a router receives a message about a topology change, it tries to find a feasible successor to the destination for which the cost metric has increased. A router sends a query to its neighbors if a feasible successor is not found. A node can be in (a) active state or (b) passive state when a query is received.

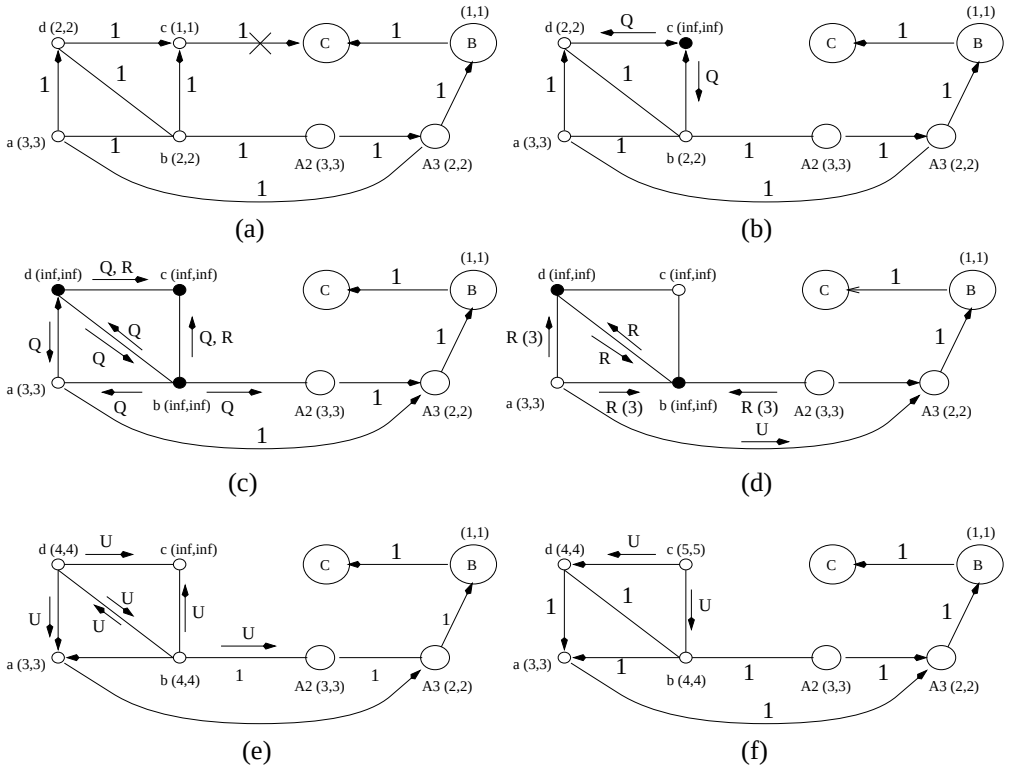

Fig. 4. Example of HIPR

Case (a): If the node is in an active state, it immediately replies to the neighbor with an infinite distance. Case (b): If the node is in a passive state, it tries to find a feasible successor for the destination and replies with the distance to destination.

Accordingly, when a query is received, a node either replies with a distance to destination or with an infinite distance. This implies, any router which is active will become passive in a finite time (because a neighbor must answer with a reply in a finite time). This implies that HIPR is free of *deadlocks* and *livelocks* in a heterarchical network. Because the areas form a virtual heterarchical network, the same results can be extrapolated to the virtual network of areas. Therefore, HIPR is free of deadlocks and livelocks for hierarchical networks. This proves the theorem. □

Theorem 3: *After a finite time $t \geq T$, the routing tables of all routers must define the final shortest path to each destination.*

Summary of Proof: Let us assume that the result is true for a stable topology at time $T(H)$ when all messages sent by routers with shortest paths having $(H-1)$ hops $(H \geq 1)$ to a given destination j have been processed by a neighbor. Also, assume that destination j is reachable through every router.

By the inductive proof for an heterarchical network [3], the result is true for HIPR within an area.

Each router maintains information about all areas and a routing table entry for a area is treated similar to an entry of a local router/destination. This forms a virtual network where each node in the network is an area and this can be viewed as a heterarchical virtual network. The rest of the proof consists of applying a similar inductive proof as in [3] to this virtual network. □

Theorem 4: *A finite time after t, no new update messages are being transmitted or processed by routers in G, and all entries in distance and routing tables are correct.*

Proof: On initialization, distance entries of distance and routing tables are set to infinity and predecessor entries are set to invalid. Since all nodes are disjoint when initialized, the table entries are correct.

Let the distance and routing table entries be correct at time t_1 and let the topology be stable at that time.

If the entries are not correct at time $t_2 = t_1 + t$, the only way a router can change its distance and routing table entries is by processing an update message. So, there could be three possibilities. A router might have processed an update, a query or a reply.

From Theorem 2, a finite time after an arbitrary change, router i must be passive. If router i receives an update for destination j from neighbor k at time t_i, the distance table of i has to be updated and routing table is updated if required. This update can be of two types: (a) for a local destination (b) for a remote destination (area)

Case (a): If the update is about a local destination, if there is no path to destination j an infinite distance is reported. If a finite distance is reported to the destination and a feasible successor is found, distance and routing tables are updated.

Case (b): If the update is about a remote destination (area), local tables are updated with the new path information.

An updated entry is added to the routing table only when the shortest path information is changed and this new information will be reported to the neighbors as routing updates.

Queries and replies are received about local destinations only (within a area). A query is always answered with a reply. The query originator, after getting all replies from its neighbors, updates the distance and routing table entries and conveys this changed information to its neighbors.

This implies that, for any given destination, a router generates new updates or queries after it reaches its final shortest path to that destination. Because every destination must obtain its final shortest path to all destinations in finite time (from Theorem 3), the theorem is true. □

5. Performance of HIPR

HIPR implements Dijkstra's shortest path algorithm over a hierarchical graph in a completely distributed manner; this means that each router works directly with the hierarchical routing tree needed to carry out part of the computation of hierarchical shortest paths. In contrast, OSPF implements Dijkstra's algorithm in a replicated manner, which means that each router needs a copy of the hierarchical graph on which it runs Dijkstra's algorithm to obtain a hierarchical routing tree of its own. Accordingly, it is expected for HIPR to outperform OSPF. To obtain insight on the average performance of HIPR in a real network, we performed simulations of HIPR and OSPF, both running over the same hierarchical network topologies. Although HIPR supports arbitrary hierarchical topologies, OSPF requires the use of a backbone that interconnects areas. Accordingly, to establish a fair basis for comparison between HIPR and OSPF, we simulated only a two-level hierarchical topology. We have simulated only those elements of OSPF that are essential for route computation. Messages are assumed to be delivered error free over an operational link and an infinite sequence number space is used in OSPF to determine the validity of link state updates.

A router receives a packet and responds to it by running the simulated routing algorithm, queueing the outgoing updates and processing the packets one at a time in the order of their arrival. If a link fails, the packets in transit are dropped. The redundant packets in the network are removed from the queue. The simulation ensures that all packets at a given simulation time are processed before the new updates are generated. The discrete-event simulation language called *Drama* [18] was used in this work.

Simulations were performed for both well-known and randomly generated topologies [12]; similar results were obtained in all cases.

The topology for which we present simulation results has ten areas and each area has ten nodes in it. Each area has at most 3 border nodes and a border node connects two or more areas. The interconnection among areas is based on McQuillan's approach to hierarchical routing. To generate backbone-based areas, we generated a random graph for each subnet; the backbone is also a random graph interconnecting these subnets. The random graphs are generated using Waxman's model [17]. Our simulation network has 100 nodes and 124 links.

To obtain the average figures, the simulation makes each link (router) in the network fail and recover, and count the steps and messages needed for each algorithm to converge. The average is then taken over all link (router) failures and recoveries. The routing algorithm was allowed to converge after each such change. In all cases, routers were assumed to provide one time unit of delay.

Router failures are modeled as the simultaneous failure of all links attached to that router. Router recoveries are modeled as all links associated with that router coming back up simultaneously. Several quantities were measured. These include, the total time elapsed for the algorithm to converge (*duration*), the total number of packets transmitted over the network; each packet may contain multiple updates (*message count*), the total number of updates and changes in the link status processed by routers (*update count*) and the total number of operations performed by each algorithm (*operations count*).

5.1 Simulation Results

For each network, we generated test cases consisting of all single failures and recoveries for both routers and links in which the routing algorithms were allowed to converge after each change. The link model allows link delay and link cost to be set independently. Each unit of time therefore represents a step in which all currently available packets are processed. Each input event is processed independently of other events received during the same simulation step. All topology changes are performed one at a time and the algorithms were allowed to converge before the next resource change.

The graphs in Figures 5–6 shows the average number of steps taken, average number of messages exchanged and the average number of operations performed for HIPR and OSPF before each algorithm converges in the case of modified Doe-Esnet topology. The error-bars in the graphs indicate the standard deviation of each of the mean values.

HIPR outperforms OSPF in all cases considered in our simulations. While the number of steps taken by HIPR to converge after a resource failure is comparable to OSPF, the number of steps taken to converge after a resource recovery is significantly better than that of OSPF. For the random topology, the number of operations required by OSPF is an order of magnitude larger than that of HIPR and the number of messages exchanged is more than twice that of HIPR for resource failure cases. This illustrates the performance advantage provided by the distributed implementation of Dijkstra's shortest path algorithm in HIPR. The average of the measured parameters is taken over all resource failures and recoveries. The algorithm was allowed to converge after each such change. Because of the distribution of values, both the mean and the standard deviation of the distribution are given. There is

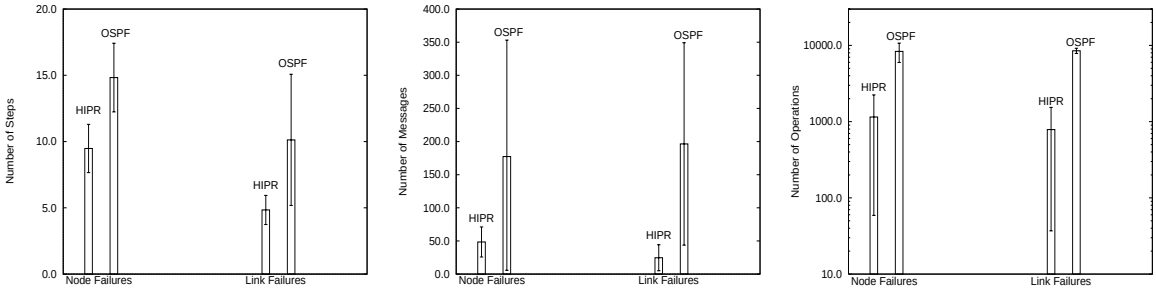

Fig. 5. Random Graph: Resource Failure

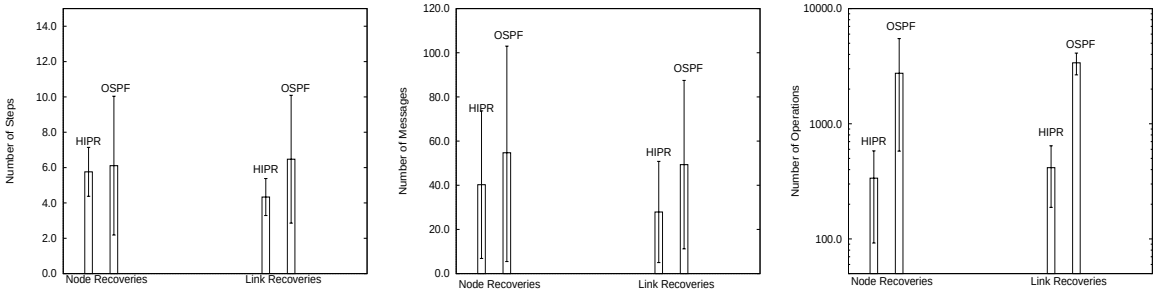

Fig. 6. Random Graph: Resource Recovery

no sampling error for the results because all possible cases are covered.

6. Conclusion

We have presented, verified and analyzed the performance of the first hierarchical routing algorithm, HIPR, based on the maintenance and exchange of hierarchical routing trees. The main idea of HIPR is to provide a distributed implementation of Dijkstra's shortest path algorithm over a hierarchical graph organized into areas. HIPR is an extension of the loop-free path-finding algorithm (LPA) using McQuillan's scheme for hierarchical routing.

The performance of HIPR was compared with that of OSPF. The simulation results presented in this paper clearly illustrate the fact that HIPR's performance is far superior to OSPF's. This suggests that an internet routing protocol based on the exchange of hierarchical routing trees would be superior to OSPF in terms of scalability and efficiency.

References

[1] A.E. Baratz and J.M. Jaffe, "Establishing virtual circuits in large computer networks", *Proc. IEEE INFOCOM 83*, April 1983, pp. 311–318

[2] C. Cheng, R. Riley, S.P.R. Kumar and J.J. Garcia-Luna-Aceves, "A loop-free extended Bellman-Ford Routing Protocol without Bouncing Effect", *ACM SIGCOMM 89*, Sept. 1989, pp. 224–236.

[3] J.J. Garcia-Luna-Aceves and Shree Murthy, "A Path-Finding Algorithm for Loop-Free Routing", *IEEE/ACM Trans. Networking*, Feb. 1997.

[4] J.J. Garcia-Luna-Aceves and N. Shacham, "Analysis of Routing strategies for packet radio networks", *Proc. IEEE INFOCOM 85*, March 1985, pp. 292–302.

[5] J.J. Garcia-Luna-Aceves and William T. Zaumen, "Area-Based Loop-Free Internet Routing", *Proc. IEEE INFOCOM 94*, April 1994, pp. 1000–1008.

[6] P.A. Humblet, "Another Adaptive Shortest-Path Algorithm", *IEEE Trans. Commun.*, Vol. 39, No. 6, June 1991, pp. 995–1003.

[7] F. Kamoun, *"Design Considerations for Large Computer Communications Network"*, PhD Thesis, University of California, Los Angeles, 1976.

[8] G.S. Lauer, "Hierarchical routing design for SURAN", *Proc. IEEE ICC 86*, June 1986, pp. 93–102.

[9] G, Malkin, "RIP Version 2 – Carrying Additional Information", RFC 1723, Nov. 1994.

[10] J. McQuillan, "Adaptive Routing algorithms for distributed computer networks", Bolt Beranek and Newman, BBN Report 2831, 1974.

[11] J. Moy, "OSPF version 2", RFC 1247, Aug. 1992.

[12] Shree Murthy, *"Routing in Packet-Switched Networks Using Path-Finding Algorithms"*, PhD Thesis, University of California, Santa Cruz, 1996.

[13] Shree Murthy and J.J. Garcia-Luna-Aceves, "A More Efficient Path-Finding Algorithm", *28th Asilomar Conference*, Nov. 1994, pp. 229–233.

[14] B. Rajagopalan and M. Faiman, "A new responsive distributed shortest path routing algorithm", *ACM SIGCOMM 89*, Sept. 1989, pp. 237–246.

[15] C.V. Ramamoorthy and W. Tsai, "An adaptive hierarchical routing algorithm", *Proc. IEEE COMPSAC 83*, Nov. 1983, pp. 93–104.

[16] J. Seeger and A. Khanna, "Reducing routing overhead in a growing DDN", *Proc. IEEE MILCOM 86*, Oct. 1986, pp. 15.3.1–15.3.13.

[17] B.M. Waxman, "Routing of multipoint connections", *IEEE J. Selected Areas in Commun.*, Vol. 6, No. 9, 1988, pp. 1617–1622.

[18] W.T. Zaumen, "Simulations in Drama", Network Information System Center, SRI International, Jan. 1991.